Object-Oriented RuleML: User-Level Roles, URI-Grounded Clauses, and Order-Sorted Terms*

Harold Boley

Institute for Information Technology – e-Business,
National Research Council of Canada,
Fredericton, NB, E3B 9W4, Canada
Harold.Boley@nrc-cnrc.gc.ca

Abstract. This paper describes an Object-Oriented extension to RuleML as a modular combination of three sublanguages. (1) User-level roles provide frame-like slot representations as unordered argument collections in atoms and complex terms. (2) URI-grounded clauses allow for 'webizing' using URIs as object identifiers for facts and rules. (3) Order-sorted terms permit typed variables via Web links into taxonomies such as RDF Schema class hierarchies, thus reusing the Semantic Web's lightweight ontologies. Besides introducing the first sublanguage with the Positional-Roled (ASCII) syntax, all three sublanguages are introduced with the OO RuleML (XML) syntax. Their semantics are sketched and their implementation paths are discussed.

1 Introduction

RuleML started in 2000 with XML-encoded positional-argument rules, and in 2002 we have introduced frame-like knowledge representation (KR) with user-level role-filler slots as unordered arguments. Since 2001 RuleML has permitted a kind of 'webizing' to allow RDF-like [LS99] KR, and RuleML 0.8 has used URIs as optional additions to, or substitutes for, individual constants as well as relation and function symbols; in the following, URI grounding will also permit URIs within clause (`fact` and `imp`) and `rulebase` labels. Finally, since 2002,

* Thanks to Michael Schroeder and Gerd Wagner for inviting me to give this RuleML'03 presentation. I also want to express my gratitude to Michael Sintek and Said Tabet for valuable contributions on several topics of this paper. Said Tabet, Benjamin Grosof, and the RuleML Steering Committee have encouraged me early on regarding the OO RuleML design. Bruce Spencer has supported the development of OO RuleML, its implementations, as well as the PR rule language. Marcel Ball and Stephen Greene gave valuable hints and performed various OO RuleML and PR syntax implementations. OO RuleML has already been employed outside the RuleML team by Virendra Bhavsar (AgentMatcher project) and Daniel Lemire (RACOFI project); further applications are being planned, e.g. by Anna Maclachlan (Metaxtract project). This research was funded by NRC as part of the Sifter project.

we have begun work on URI-based taxonomy access for order-sorted terms; this broadens the scope of webizing by typing, e.g., logic variables of RuleML via Semantic Web taxonomies such as RDF Schema (RDFS) class hierarchies.

This paper describes Object-Oriented RuleML (OO RuleML) conceived as the orthogonal combination of user-level roles, URI grounding, and order-sortedness. These orthogonal dimensions constitute three declarative OO sublanguages that can be visualized as the edges of an 'OO cube', i.e. they can be used independently from each other or can be freely combined:

The **OO contribution** of

1. **User-level roles** is to allow 'object-centered' sets of role-filler slots – much like the role-type slots of classes and role-value slots of their instances; because of the unorderedness of slot sets, the inheritance of slots will be easier than that of ordered argument sequences.
2. **URI grounding** is the provision of URIs as unique object identifiers (OIDs) for facts – much like instances – and for rules – much like methods.
3. **Order-sortedness** is making taxonomies available as declarative inheritance pathways for term typing – much like class hierarchies.

Since an ordered argument sequence can be augmented by user-level roles, the first dimension permits three choices. The other two dimensions just permit 'yes'/'no' distinctions for an entire rulebase, but even in the 'yes' case some of its clauses may be not URI-grounded or not order-sorted, since these are optional features. Because of the dimensions' mutual independence, $3 \times 2 \times 2 = 12$ modular combinations will thus be principally possible in the OO cube. Following the design rationale of RuleML 0.8 [BTW01,Bol02], these sublanguage combinations can be reflected by OO RuleML's lattice of XML DTDs/Schemas.

The sublanguages cover clauses that can be either facts or rules, where rules may be used for defining declarative methods. However, OO RuleML currently only captures the above declarative aspects of OO, not procedural aspects such as the updating of instance slots. We also omit here the possible fourth independent OO sublanguage of **signature-instantiated clauses**, which would allow the assertion of '`new`' instances.[1]

2 Object Centering via User-Level Roles

Since the beginnings of knowledge representation (KR), there have been two paradigms in this field, which will be called here *position-keyed* and *role-keyed* KR. These differ in the two natural focus points and argument-access methods of elementary representations. In predicate-centered or position-keyed KR (pKR), one predicate or relation symbol is focused, and applied to positionally

[1] As part of intertranslating between positional arguments and user-level roles there is an experimental XSLT implementation of signatures [http://www.ruleml.org/indoo]. TRIPLE [SD02] has permitted the '`new`less' generation of URI-grounded facts in rule heads.

ordered objects as arguments. In object-centered or role-keyed KR (rKR), one object identifier is focused, and associated via property roles, unordered, with other objects as arguments. Elementary representations in pKR have directly employed atomic formulas (atoms) of first-order logic, or added some syntactic sugar. Elementary representations in rKR were inspired by (Lisp) property lists and directed labeled graphs (Semantic Nets), but later have also been developed as subsets of first-order logic (Description Logic, F-logic). The more expressive representations of pKR (e.g., derivation rules) and of rKR (e.g., method definitions) maintain the different focus points. Syntactic and semantic pKR-rKR blending has been attempted, and will be demonstrated here.

In the Web, versions of both paradigms surfaced again. A kind of pKR came back with XML, because its *parent elements* are focus points 'applied to' its ordered child elements. On the other hand, a kind of rKR came back with RDF, since its *descriptions* focus a resource that has properties associating it, unordered, with other objects. Finding a common data model as the basis of Web KR has thus become a foundational issue for the Semantic Web.

In RuleML 0.8, we have used a pKR-rKR-unifying data model that generalizes the data models of both XML and RDF to express clauses (facts and rules). It is based on differentiating *type* and *role* elements in XML, where role tags (distinguished by a leading underscore) accommodate RDF properties. However, in RuleML 0.8 only *system* roles are permitted, their names cannot be taken from the application domain, and the atoms within clauses are still predicate-oriented. A pKR example will illustrate this 'system-level' solution. For this and later examples of this section we will use the Positional-Roled (PR) syntax [http://www.ruleml.org/submission/ruleml-shortation.html]. Consider a ternary `offer` relation applied to ordered arguments for the offer name, category, and price. In the PR syntax an `offer` of an 'Ecobile' can be categorized as 'special' and priced at $20000 via the following fact (Prolog-like, except that a capitalized symbol like `Ecobile` denotes an individual constant, not a variable):

```
offer(Ecobile,special,20000).
```

In RuleML 0.8 this has been marked up as follows (the `_rlab` role provides clause labels as `ind` types here and later on):

```
<fact>
  <_rlab><ind>pKR fact 1</ind></_rlab>
  <_head>
    <atom>
      <_opr><rel>offer</rel></_opr>
      <ind>Ecobile</ind>
      <ind>special</ind>
      <ind>20000</ind>
    </atom>
  </_head>
</fact>
```

Notice that the `fact` type has a `_head` role associating it with an `atom` type. The `atom`, however, uses a role, `_opr`, only for its operator association with the `rel`(ation) type. The three arguments of type `ind`(ividual) are immediate `atom` children ordered in the spirit of XML and pKR. Thus, while the `_opr` role can be moved from the prefix position to a postfix position without changing its meaning, the `ind` types are semantically attached to their relative positions. This fact representation thus requires users and applications (e.g., XSLT) to 'store' the correct interpretation of the three arguments separately, and any extension by additional arguments requires changes to these positional interpretations, except when new arguments are always added at the (right) end only.
A 'user-level' solution in the spirit of RDF and rKR thus is to introduce (user-level) roles `name`, `category`, and `price` for the arguments. Our `offer` can then be represented in the PR syntax as follows (inspired by F-logic [KL89]):

```
offer(name->Ecobile;category->special;price->20000).
```

In OO RuleML this can be marked up thus:

```
<fact>
  <_rlab><ind>rKR fact 1</ind></_rlab>
  <_head>
    <atom>
      <_opr><rel>offer</rel></_opr>
      <_r n="name"><ind>Ecobile</ind></_r>
      <_r n="category"><ind>special</ind></_r>
      <_r n="price"><ind>20000</ind></_r>
    </atom>
  </_head>
</fact>
```

Actually, a single (system-level) metarole `_r` is employed here with different (user-level) values of the XML attribute `n`(ame). The XML DTDs/Schemas of RuleML thus only require a small change to introduce rKR for RuleML's atomic formulas [http://www.ruleml.org/indoo]. The correct interpretation of the three arguments is no longer position-dependent and additional arguments such as `expiry` and `region` can be added without affecting any existing interpretation.

Now suppose we wish to keep the earlier pKR for the first three (mandatory) arguments, but use rKR for two extra (optional) arguments. For this, we can employ a blend of one ordered sequence of arguments plus a role-keyed set of arguments before and/or after:

```
offer(Ecobile,
      special,
      20000;
      expiry->2003-12-31;
      region->North America).
```

```
<fact>
  <_rlab><ind>prKR fact 1</ind></_rlab>
  <_head>
    <atom>
      <_opr><rel>offer</rel></_opr>
      <ind>Ecobile</ind>
      <ind>special</ind>
      <ind>20000</ind>
      <_r n="expiry"><ind>2003-12-31</ind></_r>
      <_r n="region"><ind>North America</ind></_r>
    </atom>
  </_head>
</fact>
```

Since the XML DTDs/Schemas of OO RuleML naturally extend those of RuleML 0.8 by optional sets of roles around the possibly empty argument sequence, such pKR-rKR blends are implicitly taken care of.

As an example of a binary rKR relation, the PR syntax and OO RuleML can represent customer Peter Miller's gold status thus:

```
customer(name->Peter Miller;status->gold).
```

```
<fact>
  <_rlab><ind>rKR fact 2</ind></_rlab>
  <_head>
    <atom>
      <_opr><rel>customer</rel></_opr>
      <_r n="name"><ind>Peter Miller</ind></_r>
      <_r n="status"><ind>gold</ind></_r>
    </atom>
  </_head>
</fact>
```

Notice that the meaning of such unqualified roles is *local* to their atoms: within `offer` objects, `name` refers to offer names; within `customer` atoms, to customer names. Again, further roles could be easily added.

Variables can be introduced in all of these versions. In the PR syntax, variables are prefixed by a "`?`"; in OO RuleML, they use `var` type tags instead of `ind` markup. This, then, permits the representation of rules for both pKR and rKR.

As an example, here is an rKR version of a `discount` rule in PR syntax (the `price` role with the anonymous filler variable "`_`" requires the presence of price information but does not use it – without that requirement an anonymous rest-role variable could have been employed instead):

```
discount(offer name->?off;
         customer name->?cust;
         awarded amount->10)
   :-
offer(name->?off;
      category->special;
      price->_),
customer(name->?cust;
         status->gold).
```

In OO RuleML this can be marked up as follows:

```
<imp>
  <_rlab><ind>rKR rule 1</ind></_rlab>
  <_head>
    <atom>
      <_opr><rel>discount</rel></_opr>
      <_r n="offer name"><var>off</var></_r>
      <_r n="customer name"><var>cust</var></_r>
      <_r n="awarded amount"><ind>10</ind></_r>
    </atom>
  </_head>
  <_body>
    <and>
      <atom>
        <_opr><rel>offer</rel></_opr>
        <_r n="name"><var>off</var></_r>
        <_r n="category"><ind>special</ind></_r>
        <_r n="price"><var/></_r>
      </atom>
      <atom>
        <_opr><rel>customer</rel></_opr>
        <_r n="name"><var>cust</var></_r>
        <_r n="status"><ind>gold</ind></_r>
      </atom>
    </and>
  </_body>
</imp>
```

Notice that this `imp`(lication) rule derives a `discount`-centered `atom` in its head from the following conjunction in its body: An `offer`-centered query asks for `category` = `special` (but masks the specific `price` with an anonymous variable) and a `customer`-centered query asks for `status` = `gold`. The `off` and `cust` variable bindings obtained under `name` in the body are differentiated as `offer name` and `customer name` in the head and a 10% discount is awarded. Using rKR fact 1 and rKR fact 2, rKR rule 1 derives this amount for an Ecobile purchase by Peter Miller.

The **semantics** of rKR's clause sets with user-level roles can be defined by explaining how pKR's (here, LPs [Llo87]) notions of clause instantiation and ground equality (for the model-theoretic semantics) as well as unification (for the proof-theoretic semantics) should be extended.

Since rKR role names are assumed here to be non-variable symbols, *rKR instantiation* can recursively walk through the fillers of user-level roles, substituting dereferenced values from the substitution (environment) for any variables encountered.

Since OO RuleML uses explicit rest variables, *rKR ground equality* can recursively compare two clauses after lexicographic sorting – w.r.t. the role names – of the role-filler slots of atoms and complex terms.

Since OO RuleML uses at most one rest variable per atom or complex term, *rKR unification* can perform sorting as in the above ground equality, use the above rKR instantiation of variables, and otherwise proceed left-to-right as for pKR unification, pairing up identical roles before recursively unifying their fillers.

The **implementation** of OO RuleML for rKR has been done both via an XSLT translator to positional RuleML and via an extension of the Java-based jDREW interpreter [Spe02]. This has already been used for representing product-seaking/advertising trees in the tree-similarity-based AgentMatcher system [BBY03] and for expressing music filtering rules in the collaborative e-learning system RACOFI [ABB+03].

3 URI Grounding of Clauses

Our previous rKR clauses did not use object identifiers (OIDs) for providing each object with an (object) identity. RDF has introduced a new flavor of OIDs for describing resources via their URIs.[2] This style of KR, here called URI-grounded KR (gKR), is also possible in OO RuleML by permitting a `wid` (`w`eb `id`) attribute within the `ind` type of an `_rlab` (`r`ule `lab`el) or – not further detailed here – of an entire `_rbaselab` (`r`ule`base lab`el); this is complemented by a `widref` (`w`eb `id ref`erence) attribute within the `ind` type of a referring slot filler; so, `wid` and `widref` are dual like XML's `id` and `idref` and RDF's `about` and `resource`.[3]

The previous rKR fact 1 can thus be URI-grounded such that it is specialized to grKR fact 1 for *the* Ecobile occurring as offer 37 in a certain catalog as shown below. Similarly, the `ind` type of `Ecobile` can have a `widref` to another grounded rKR fact.

[2] While RDF was originally mostly intended for 'metadata' describing 'data' at URI resources, it has meanwhile been taken up for general KR where – even besides RDF's "blank nodes" – none of these URIs need link to 'data'. But still RDF's OIDs are 'about' objects, not (URI) addresses of objects: Usually several RDF descriptions are collected in a document that itself can be located at a single URI unrelated to the URIs being described.

[3] With OIDs, circular references become possible in OO modeling, RDF, OO RuleML, and XML, but these can often be statically detected to avoid (indirectly) self-referential clauses and the well-known logical paradoxes. Our semantics will perform a graph search with loop detection for obtaining a closure of objects.

The other grounded rKR fact, grKR fact 3, uses a different kind of URI attribute within an `ind`: `href` refers to a 'home page' characterizing the individual (e.g., `gas`). Generally, while `widref` presupposes that a description about the URI exist but not that the URI (currently) exist, `href` presupposes that the URI (currently) exist but not that a description about the URI exist.

Finally, also *global* user roles can be constructed as Qualified Names (QNames) whose qualifier is a namespace prefix, e.g. `s` or `t`, which is associated with a URI in the namespace declaration of the `rulebase` type that surrounds all RuleML clauses. Fragment identifiers ("`#`") are employed to point into the URI-addressed document.

```
<ruleml:rulebase
 xmlns:ruleml="http://www.ruleml.org/dtd/0.83/ruleml-oodatalog.dtd"
 xmlns:s="http://offercore.org/offerproperties#"
 xmlns:t="http://productcore.org/productproperties#">
 <fact>
  <_rlab><ind wid="http://catalist.ca/37">grKR fact 1</ind></_rlab>
  <_head>
    <atom>
      <_opr><rel>offer</rel></_opr>
      <_r n="s:name"><ind widref="http://ecobile.com">Ecobile</ind></_r>
      <_r n="s:category"><ind>special</ind></_r>
      <_r n="s:price"><ind>20000</ind></_r>
    </atom>
  </_head>
 </fact>
 <fact>
  <_rlab><ind wid="http://ecobile.com">grKR fact 3</ind></_rlab>
  <_head>
    <atom>
      <_opr><rel>product</rel></_opr>
      <_r n="t:name"><ind">Ecobile SX</ind></_r>
      <_r n="t:fuel"><ind href="http://naturalgas.org">gas</ind></_r>
      <_r n="t:horsepower"><ind>90</ind></_r>
      <_r n="t:displacement"><ind>1550</ind></_r>
    </atom>
  </_head>
 </fact>
</ruleml:rulebase>
```

The use of QNames in an attribute value such as the above `s:name` in `n="s:name"` has been discussed in a recent W3C TAG Finding [http://www.w3.org/2001/tag/doc/qnameids-2002-06-04].

These OO RuleML facts – except for their optional labels (e.g., grKR fact 1), their explicit relation names (e.g., `offer`),[4] and their both named and grounded

[4] RDF's anonymous grounded descriptions can be represented by substituting a named relation markup such as `<rel>offer</rel>` with an anonymous one such as `<rel/>`, keeping the `_rlab`-embedded grounding `<ind wid="http://catalist.ca/37">`.

arguments (e.g., `Ecobile` on http://ecobile.com)[5] – correspond to the following RDF descriptions:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://offercore.org/offerproperties#"
  xmlns:t="http://productcore.org/productproperties#">
  <rdf:Description about="http://catalist.ca/37">
    <s:name rdf:resource="http://ecobile.com"/>
    <s:category>special</s:category>
    <s:price>20000</s:price>
  </rdf:Description>
  <rdf:Description about="http://ecobile.com">
    <t:name>Ecobile SX</t:name>
    <t:fuel>gas</t:fuel>
    <t:horsepower>90</t:horsepower>
    <t:displacement>1550</t:displacement>
  </rdf:Description>
</rdf:RDF>
```

Like RDF properties, OO RuleML roles can thus be based on RDF Schema definitions (e.g., `subPropertyOf`) at the documents pointed to by their qualifier URIs; the next section will further expand on this kind of RDF-RuleML interoperation (for OO RuleML sorts, using `subClassOf`). However, the following is also important when comparing RDF and OO RuleML:

1. RDF's domain-specific properties, e.g. `s:name`, `s:category`, and `s:price`, are used as XML elements, so RDF serializations cannot be given a generic DTD/Schema; OO RuleML's corresponding metarole `_r` contains the domain-specific roles only as XML attribute values, so is amenable to a generic DTD/Schema (actually, this is a simple extension to RuleML's generic DTD/Schema [http://www.ruleml.org/indoo]).
2. For (ground) facts, OO RuleML representations contain little more information than their RDF counterparts: RDF diagrams can be regarded as the minimal basic data model; OO RuleML could thus act as an alternate RDF serialization amenable to DTD/Schema validation, hence to embedding into other valid XML (including XHTML) documents.
3. Once RDF facts are captured in OO RuleML, RDF rules (and queries) over them are directly available as well: they can be taken from OO RuleML's system of sublanguages.
4. Issues and solutions can be transferred between RDF and RuleML already now (as started in [http://lists.w3.org/Archives/Public/www-rdf-rules]): some candidates are the semantic and implementation issues raised below.

[5] RDF's grounded-only objects used as slot fillers can again be represented by substituting a named, grounded argument markup such as `<ind widref="http://ecobile.com">Ecobile</ind>` with a grounded-only one such as `<ind widref="http://ecobile.com"/>`.

URI grounding is orthogonal to user-level roles. Actually, grounding has been used for pKR in RuleML 0.8. For example, our initial pKR fact 1 can be URI-grounded as follows:

```
<ruleml:rulebase
  xmlns:ruleml="http://www.ruleml.org/dtd/0.83/ruleml-oodatalog.dtd"
  <fact>
   <_rlab><ind wid="http://catalist.ca/37">gpKR fact 1</ind></_rlab>
   <_head>
     <atom>
       <_opr><rel>offer</rel></_opr>
       <ind widref="http://ecobile.com">Ecobile</ind>
       <ind>special</ind>
       <ind>20000</ind>
     </atom>
   </_head>
  </fact>
 </ruleml:rulebase>
```

The XML DTDs/Schemas of RuleML only require a small change to introduce gKR by allowing `wid`/`widref` attributes on RuleML's `ind` and `cterm` elements. Their validation could be defined similarly to XML's `id`/`idref` validation (after URI normalization), requiring unique `wid` values and one `wid` value to exist for every `widref` value of a local document.[6]

The **semantics** of gKR's URI grounding in OO RuleML can be divided into three parts.

First, URI strings are often processed by rules for expansion, redirection, etc. (by a "canonicalization algorithm" [http://www.ietf.org/rfc/rfc2396.txt] [BLFM98]) before their referenced 'Web objects' (e.g., Web documents) can be retrieved or they turn out to be 'broken links'. Hence, it appears natural to check for semantic URI equality using a notion of string rewriting [http://www.loria.fr/~vigneron/RewritingHP]. We regard two URIs, which may be syntactically different, as semantically equal iff they are processed or rewritten to the same (normal form) URI just before both link to a (namely, the same) Web object or both lead to a broken link error. A *URI normal form* is a URI string that cannot be rewritten any further but either directly refers to a Web object or directly leads to a broken link error. For a gKR rulebase B we consider – at any given time t – a *URI rewriting system* $(s(B), R)$ over the finite set $s(B)$ of URIs used for the grounding of B. The rewriting relation R contains URI

[6] For *distributed* grKR, where several documents contribute to describe the same URI, the situation becomes similar to RDF's `about`/`resource`: `wid` values will not be globally unique and one or more facts with the same `wid` value can exist in separate documents for every `widref` value. The unorderedness of rKR would permit to 'materialize' all distributed '`wid`-equal' facts into a single virtual one on demand, restoring the original centralized situation. For distributed gpKR the orderedness of pKR would prevent this virtual 're-centralization'.

expansion rules such as for extending certain URIs by "`/`", "`index.html`", etc. R also contains redirection rules for replacing entire URIs by other URIs. For the grounding semantics the URI rewriting system must be *convergent* (terminating and confluent), i.e. unique normal forms must exist. Testing the syntactic equality of these normal forms can then be used to check for the semantic equality of any pair of URIs used in grounding. This semantic equality will be needed in the unification of URI-grounded terms as well as in the other parts below.

Second, a `wid` attribute within the `_rlab` of clauses or within the `_rbaselab` of rulebases semantically labels these elements with the normal form of the URI string of the attribute's value. On the other hand, `widref` attributes inside a clause semantically initiate the following graph search for the *clause closure* in the current document:[7] Retrieve the clause or rulebase having the same `wid` normal form label as exhibited by a `widref` attribute's value; recursively continue retrieval with all the `widref` attributes of all the clauses retrieved directly or within rulebases – just ignoring duplicates resulting from circular references – until a fixpoint is reached. The URI grounding semantics of the original `widref`-attributed clause then is the clause closure of all these retrieval results (conversely, for circular references, elements of this set can contain the current clause in the set of *their* grounding retrieval results). The URI grounding semantics of a rulebase is the union of the clause closures of all its clauses.

Third, an `href` attribute inside a clause semantically – at any given time t – makes the normal form of the attribute's URI value link to the semantics of the Web object or, for a broken link, causes it to denote an error object.[8] If the Web object linked to is another gKR rulebase, *its* semantics can be obtained as described in the current section; similarly, for the semantics of sections 2 and 4; further Semantic Web objects (e.g., in OWL [DS03]) could be covered as well.

Since the first part of this gKR semantics augments equality and unification, it can be used to extend the pKR and rKR semantics of section 2. The second part, constructing an additional clause closure, can be modularly added to the pKR/rKR semantics. Similarly, the semantics of the third part is only linked to, hence completely decoupled from, the pKR/rKR semantics.

An **implementation** of OO RuleML for gKR has not been attempted yet, mainly because it depends on an improved specification of semantic URI equality rules [http://www.w3.org/2001/tag/ilist#URIEquivalence-15] for the URI rewriting system.

---

[7] Although for non-distributed KR `wid` and `widref` act (internally) within a document only, the rewriting described in the first part is still performed (externally) using Web normal forms of the URIs. This will simplify the maintenance of large gKR documents and ensure later interoperability of gKR documents distributed over different namespaces.

[8] Currently, `href` and `type` (see next section) are the only OO RuleML attributes whose meaning links into the Web, but several related attributes of this kind could be distinguished.

4 Term Typing via Order-Sorted Taxonomies

Terms, in particular variables, in the previous pKR/rKR and gKR clauses are still untyped using *unsorted* KR. We proceed here to *order-sorted* KR (sKR) that is based on a special treatment of sort predicates and sorted individuals, variables, etc. in clauses. With sort restrictions directly attached to variables, hence usable during unification, proofs can be kept at a more abstract level, thus reducing the search space. An independently defined sort hierarchy, e.g. in RDFS (using `subClassOf`) or OWL, can be employed as the taxonomy that constitutes the partial order of the resulting order-sorted logic. We have developed a webized construct for linking RuleML variables to such externally defined sort hierarchies of the Semantic Web.

The way sorted RuleML variables link to RDFS classes is the following:

- Basically, the class hierarchy of an order-sorted logic – e.g. in RDFS – can be accessed from RuleML in a similar way as it can from RDF.
- RDF's use of `rdf:type` for taxonomic RDFS typing of individuals/resources is transferred to RuleML's typing of `ind`s.
- Additionally, we propose a new RDFS use: to access unchanged RDFS for typing RuleML variables, noting that the RDFS taxonomy must then be cycle-free (and, if we use an OWL taxonomy, it must also be consistent).

The technical construct is again based on namespace declarations using fragment identifiers ("`#`") to point into the RDFS document containing the *class* definition to be used as a type. This `#`*class* is assumed to exist there, usually with one (or more) `subClassOf` relations defining (multiple) inheritance. An `ind` or `var` is then typed via a `type` attribute augmenting the namespace prefix by the *class* name. Our earlier rKR rule 1 can thus be typed as follows (with *class* `Offer` and *class* `Customer`):

```
<ruleml:rulebase
 xmlns:ruleml="http://www.ruleml.org/dtd/0.83/ruleml-oodatalog.dtd"
 xmlns:t="http://distribcore.org/distribclasses#"
 xmlns:u="http://customercore.org/custclasses#">
 <imp>
  <_rlab><ind>rsKR rule 1</ind></_rlab>
  <_head>
    <atom>
      <_opr><rel>discount</rel></_opr>
      <_r n="offer name"><var type="t:Offer">off</var></_r>
      <_r n="customer name"><var type="u:Customer">cust</var></_r>
      <_r n="awarded amount"><ind>10</ind></_r>
    </atom>
  </_head>
```

```
  <_body>
    <and>
      <atom>
        <_opr><rel>offer</rel></_opr>
        <_r n="name"><var type="t:Offer">off</var></_r>
        <_r n="category"><ind>special</ind></_r>
        <_r n="price"><var/></_r>
      </atom>
      <atom>
        <_opr><rel>customer</rel></_opr>
        <_r n="name"><var type="u:Customer">cust</var></_r>
        <_r n="status"><ind>gold</ind></_r>
      </atom>
    </and>
  </_body>
 </imp>
</ruleml:rulebase>
```

Again, the use of QNames in an attribute value such as the above `t:Offer` in `type="t:Offer"` has been recently discussed in [http://www.w3.org/2001/tag/doc/qnameids-2002-06-04].

Here we assume that the URI `http://distribcore.org/distribclasses` links to an RDFS document containing a definition of `Offer`, e.g. specifying it as a subclass of `Distribution`. Similarly, for the URI `http://customercore.org/custclasses`. Sorted unification of two typed variables can then employ the RDFS sort hierarchy to find the greatest lower bound (glb) of their types, failing if it does not exist. For example, suppose `Sale` is defined as a subclass of both `Offer` and `Promotion`, another subclass of `Distribution`.

Based on this, the OO RuleML query

```
<ruleml:query
  xmlns:ruleml="http://www.ruleml.org/dtd/0.83/ruleml-oodatalog.dtd"
  xmlns:v="http://distribcore.org/distribclasses#">
  <_rlab><ind>rKR query 1</ind></_rlab>
  <_body>
    <atom>
      <_opr><rel>discount</rel></_opr>
      <_r n="offer name"><var type="v:Promotion">prom</var></_r>
      <_r n="customer name"><ind>Peter Miller</ind></_r>
      <_r n="awarded amount"><var>Rebate</var></_r>
    </atom>
  </_body>
</ruleml:query>
```

will unify with the head of the above defined rsKR rule 1 by binding `<var type="v:Sale">prom</var>` to `<var type="t:Sale">off</var>`, where

`Sale` is found in the RDFS document as the glb of `Offer` and `Promotion`, and also binding `<var type="u:Customer">cust</var>` to `<ind>Peter Miller</ind>` and `<var>Rebate</var>` to `<ind>10</ind>`.

Besides the above combination of sKR with rKR, sKR can also be combined with gKR, hence with grKR, and furthermore with the pKR versions.

The XML DTDs/Schemas of RuleML only require the following change for sKR: the introduction of a `type` attribute on `ind`, `var`, and `cterm` elements.

The **semantics** of sKR could be given directly but can also be reduced to unsorted KR in a well-known manner discussed here for sorted terms that are variables. All occurrences of a sorted variable are replaced by their unsorted counterparts plus a body-side application of a sort-corresponding unary predicate to that variable (sorted facts thus become unsorted rules). Moreover, the definition of the unary predicate reflects the subsumption relations of the sort taxonomy via rules.

Independently of using this reduction, the sKR semantics can be combined with the pKR/rKR and gKR semantics of sections 2 and 3 in a modular fashion.

The **implementation** of sKR has been performed directly (without the above reduction) for various sorted Prolog pKR systems before RDFS became available as a Web-based taxonomy language. We plan to adapt sorted indexing techniques for Prolog to RDFS and to the Java-based implementation of the Fredericton OO jDREW interpreter for OO RuleML.

5 Conclusions

Object-Orientation in OO RuleML currently comprises object-centered user-level roles, object identifiers for URI-grounded clauses, and class hierarchies over order-sorted terms. While these OO sublanguages can be used as three independent extensions to RuleML 0.8, they can also be modularly combined, pairwise or "all in one". A fourth conceivable OO sublanguage consists of signature declarations – possibly roled, grounded, and/or sorted – and their instantiation to '`new`' clauses (normally facts). However, the latter would cross the borderline between declarative KR, currently focussed in RuleML, and procedural KR; this borderline has already been touched with URI-grounded clauses.

The principal relationship between OO RuleML and OO Programming is as follows: Clauses that are (ground) facts correspond to instances, signatures can be viewed as classes, and rules may be used for defining methods. Again, in OO RuleML, those methods are declarative, for querying or deriving information, not procedural as in OOP, for updating a knowledge base or a program state.

There are several connections between OO RuleML and other RuleML extensions, of which we mention those related to some RuleML Technical Group (TG):

- The ongoing work on production and reaction rules often uses – as in Jess – instances/facts stored in the CLIPS format, which employs user-level roles. Moreover, much of the effort in the **Reaction Rules** TG utilizes object-oriented modeling in the style of OMG's UML, OCL, and MOF, whose integration has become easier with OO RuleML (similarly for events).
- The efforts in the **Ontology Combination** TG have led among other results to Description Logic Programs [GHVD03], which can be represented employing user-level roles and order-sorted terms.
- The TG on **Frames, Objects, and RUle Markup (FORUM)** [http://forum.semanticweb.org] – based mostly on F-logic [KL89] and TRIPLE [SD02] – has started studying rules for RDF and graph-based data, which can be mapped to roled, grounded, and possibly sorted OO RuleML.

OO RuleML for rKR has been first implemented via a positionalizing XSLT translator to jDREW for pKR [Spe02], and then directly in the form of the Java-based OO jDREW [http://www.ruleml.org/indoo]. Further tools for OO RuleML currently available from this home page include a positionalizing XSLT translator for OO RuleML's role-weighted extension as well as a pair of Java-based translators between the PR syntax and OO RuleML for rKR and pKR [http://www.ruleml.org/submission/ruleml-shortation.html].

OO RuleML has already served as an interchange format in two major applications. In the RACOFI system [http://racofi.elg.ca] OO RuleML rules are utilized in conjunction with collaborative-filtering techniques for querying a database of music objects rated in multiple dimensions [ABB+03]. For the Treesim algorithm [http://www.cs.unb.ca/~boley/treesimilarity] the role-weighted OO RuleML extension is utilized to represent all product-seaking/advertising trees of the AgentMatcher system [BBY03].

References

[ABB+03] Michelle Anderson, Marcel Ball, Harold Boley, Stephen Greene, Nancy Howse, Daniel Lemire, and Sean McGrath. RACOFI: A Rule-Applying Collaborative Filtering System. Submitted for publication, August 2003.

[BBY03] Virendra C. Bhavsar, Harold Boley, and Lu Yang. A Weighted-Tree Similarity Algorithm for Multi-Agent Systems in e-Business Environments. In *Proc. Business Agents and the Semantic Web (BASeWEB) Workshop*, pages 53–72. NRC 45836, June 2003.

[BLFM98] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. Request for Comments 2396, Network Working Group, The Internet Society, August 1998.

[Bol02] Harold Boley. The Rule Markup Language: RDF-XML Data Model, XML Schema Hierarchy, and XSL Transformations. In *Proc. 14th International Conference on Applications of Prolog (INAP2001)*. The University of Tokyo, LNAI 2543, Springer-Verlag, October 2002.

[BTW01] Harold Boley, Said Tabet, and Gerd Wagner. Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In *Proc. Semantic Web Working Symposium (SWWS'01)*, pages 381–401. Stanford University, July/August 2001.

[DS03] Mike Dean and Guus Schreiber. OWL Web Ontology Language – Reference. W3C Candidate Recommendation, W3C, August 2003.

[GHVD03] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proc. 12th Intl. Conf. on the World Wide Web (WWW-2003)*. Budapest, Hungary, May 2003.

[KL89] Michael Kifer and Georg Lausen. F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and Scheme. In James Clifford, Bruce G. Lindsay, and David Maier, editors, *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 134–146, Portland, Oregon, 31 May–2 June 1989.

[Llo87] John W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, Heidelberg, New York, 1987.

[LS99] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Recommendation REC-rdf-syntax-19990222, W3C, February 1999.

[SD02] Michael Sintek and Stefan Decker. TRIPLE – A Query, Inference, and Transformation Language for the Semantic Web. In *1st International Semantic Web Conference (ISWC2002)*. Sardinia, Italy, June 2002.

[Spe02] Bruce Spencer. The Design of j-Drew: A Deductive Reasoning Engine for the Web. In *Joint CoLogNet Workshop on Component-based Software Development and Implementation Technology for Computational Logic Systems of LOPSTR '02*. Technical University of Madrid, Spain, September 2002.